**Team Awesome += Five’s High Speed Cache**  
D. Grosvenor, J. Gaither, and T. Crumbley  
ECE 432 – Fall 2007  
University of Virginia  
[drg4m@virginia.edu](mailto:drg4m@virginia.edu), [jtg4r@virginia.edu](mailto:jtg4r@virginia.edu), [trc5r@virginia.edu](mailto:trc5r@virginia.edu)

**ABSTRACT**The purpose of this paper is to present the design of the high speed cache developed by Team Awesome += Five for Portable Instruments Company (also known as PICo). The high speed cache will be used during periods of high data transfer, during encryption and compression. The topics within this paper include a description of the design process as well as an overview of the components developed and used within the cache. This paper will also address the problems that arose throughout the design process, along with the methodologies the design team used in order to debug said problems.

1. **INTRODUCTION**

As scaling decreases in the integrated circuit (IC) industry, demand for high speed electronics are ever increasing, and present a major challenge to develop high speed components. To further increase this challenge, a trade off presents itself such that a decrease in the size of the IC increases the complexity of the device. The metal that interconnects components within a circuit becomes more layered and complex, further increasing the resistance and capacitance of the circuitry. Variables that were negligible years ago are now becoming increasingly important.

The challenges described above present opportunities for circuit designers to work for companies that require this expertise to develop complex circuitry. This is why the design team Awesome += Five (A+=5) decided to take the opportunity to obtain a contract from PICo to develop a static random access memory (SRAM). A SRAM is a memory device that can read or write to any address, regardless of where it previously accessed. Also, since it is static memory, the bits are volatile and lose their values when disconnected from power. Out of two different types of SRAM to choose from, the team decided on the high speed cache. A high speed cache is typically used to temporarily store data that will likely be used again soon, or in between two levels of memory that would otherwise be much slower. PICo requested a high speed memory that can be used in their new mobile communications technology. The cache will only be used when transferring data at high speeds during the encryption and compression phase built into the mobile technology. This factors into the team’s metric, causing it to focus more on the total delay than power or area. The total delay is the worst case delay of either the read or the write operation.

The design process of the cache was a long and difficult process that was also educational. In order to accomplish such a large task, the team split the process into smaller tasks. The first task that the team accomplished was discussing and figuring out what they actually wanted in their cache array. After planning what to design on a block level, the team started to build the circuitry behind the memory blocks, and most importantly, the bit cell.

The bit cell is the most important part of the SRAM, because these devices store the value of the bits in the cache. Regarding the bit cell, the team ran many tests in order to assess what transistor sizings to use in order to prevent read and write upsets, along with ensuring robustness. Once the team performed an adequate amount of testing, they moved onto integrating all the blocks in the array together.

Regarding testing, the majority of time spent by the team was spent on testing the circuitry of the SRAM. Before going onto the next step after designing each part of the array, the team spent time trying to prove that the newest addition to the circuit worked and did not upset the logic of the memory. This involved looking at the delays, evaluating bit cells at the process corners, and other circuit evaluation techniques. The team used methodologies they learned in their academic careers as well as new ones they developed throughout the design process of this project.

1. **METHODS**

**2.1 Design Process**

*2.1.1 Problem*

In order to deliver a high quality cache to PICo, the design team had to develop a good understanding of the problem they were facing. To help with this, Professor Ben Calhoun served as a liaison between PICo and the design team; he helped clarify any misunderstanding the design teams may have had regarding the deliverables.

The main issue regarding the deliverable was the constraints and metrics that the teams had to adhere to. For the high speed cache the total capacity is 64kb, and the main metric is the total power \* delay ^ 2 \* Area. Furthermore, this word size had to be 32 bits long, and had to take in and output thirty two bits. Thankfully, we did not have to worry about generating read, write, or clock signals as they were all given to us.

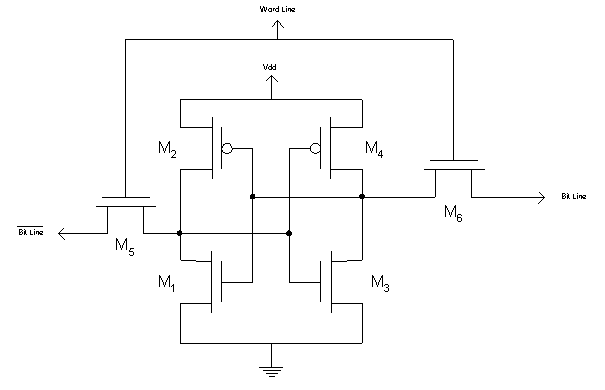


Figure : SRAM Bit cell

The project team took this information, and decided to initially brainstorm ways to create a memory that followed these constraints and optimize the metrics, specifically the delay. The team decided that 2 words per row in a block would be a good idea, as this would provide several benefits. First, since we were designing a high speed cache, we were worried about resistance and interconnect capacitance. We figured that by having two words per row, this would decrease the resistance created by the wires. Also by using two words per row, it requires an address bit to decode which one. Because of this, we had one less address bit that had to go through a decoder, improving the speed and decreasing the capacitance. We also believe that since we wanted 64 bits per block, it would put less stress on bit lines and the inputs of the circuit, allowing for improved performance.

*2.1.2 Bit cell and Initial Block Design*

After agreeing as to what we wanted in our memory, we drew out block schematics for what one block of the memory would look like, thus beginning the design process of our circuit. The purpose of drawing a block diagram before actually creating any circuitry was to have an idea of where we were going before getting too far in and then have to redo a lot of our work.

With the rough schematic of the memory block, we began creating schematics for the bit cell and surrounding logic necessary to access it. Referring to Figure 1, the bit cell consists of a total of six transistors, four transistors for the 2 inverters (M2/M1 and M4/M3) that feed into each other, and 2 NMOS access transistors (M5 and M6) connected to the inverters. Since the two inverters are feeding into each other, they are holding the values at their intermediate nodes, which are connected to the drain of each access transistor. The access transistors are there to ensure that the loop is only accessed when there is a read or a write. The most difficult thing in designing the bit cell is setting the sizes of the transistors to prevent read upsets. A read upset occurs when the value inside the bit cell is flipped during a read, causing an error. These most typically occur due to poor sizing, so as a designer its their responsibility to ensure proper sizing to prevent this. Furthermore, the bit cell had to be tested to account for all the different types of variation to ensure correctness. The types of variation that we were worried about were global variation, which happens during the manufacturing process; variation with temperature and voltage. By testing at all these regions and adjusting the bit cell appropriately, we were able to show that it works.

Also within the block are all the pieces that interact with the bit cell. Before testing the blocks in an array environment, we used the Analog Environment in cadence to generate a lot of our signals. Also, the only real circuits we had in the block were the address decoders and the sense amps. Since we used one of the Address bits to choose the word, we only needed to put the six least significant through the decoder. We originally planned on using a dynamic decoder due to them being faster than a static decoder; however, after much testing and integration, the static decoder seems to pay off more than the dynamic. This is because the dynamic was harder to get to work correctlt, and the sizing of the transistors just did not seem to work properly. The other main problem with the dynamic decoder was that we had to take into account that the dynamic decoder had to not break and be functional for all clocks, and problems arose with a stop clock (which means that the frequency of clock is zero, and the period of the clock is inifinity). Furthermore, in order to decide which one to choose, we ran the same simulations on both, and tried to see which one was fastest. In our simulations the dynamic decoder never seems to drastically out-perform the static SRAM.

The other major circuitry that we use near the bit cells is the differential voltage sense amplifier. The sense amplifier is very important to the operation of the memory; it does not change any of the functionality of the memory, but instead makes it much more useable. In an array of many SRAM cells, the resistance and interconnect capacitance is much larger than it would be, which causes the signal that comes out of a read to change much slower, and therefore has to take much longer to see a voltage swing. Without the sense amp in the circuit, the Read operation would be a major bottleneck, and therefore the clock would have to become slower to compensate. However, with the sense amp in place, the performance of the circuitry improves. The sense amp amplifies the difference between the bit line and the bit line bar, and uses that to show a full swing read. In essence, the sense amplifier helps reduce the delay times and the power dissipation in the SRAM. However, with all these benefits, the sense amp has probably been the hardest design challenge for the team.

There are several design issues to be concerned about regarding the sense amplifier. With the type of sense amp that the team decided the use, the amplifier has to be symmetric in respect to the y-axis, such that the bit line and bit line bar are pulled down proportionally the same. Without the symmetry, the sense amp may not adjust the voltage at one end down correctly and the correct output will not be read. Another thing about the sense amp is that despite the fact that nothing is wired at one end of the sense amp, it still needs the same buffers that come out the output end. This supports the fact that there needs to be symmetry across the y-axis for the correct signal to be output. Also, the NMOS transistors should be sized bigger than the PMOS because they need to pull the bit lines down to ground. However, the largest impact on the effectiveness of the sense amp is the timing of the signals. In order for the sense amp to properly latch, several things have to happen during the read. After the precharge, and the bit lines are both at voltage high, word line then has to go from low to high. This turns the access transistors on, and allows the bit lines to float. The important part of the concept of integrating the sense amp into the rest of the signal is that shortly after word line goes high, sense amp enable should go high. The significance of this is that the sense amp works quite well when it senses a small delta in voltage, so the sense amp enable signal should be generated as though it is a delayed word line. Shortly there after, once the signals are all stable, the word line can be cut off, stabilizing the bit lines. After that, the output of the sense amp needs to be latched to a register connected to the output before the bit lines precharge again. Thus, when precharge goes active again, it will not draw as much power as it would when one of the bit lines would be driven low. The sense amplifier is a powerful tool to use in the SRAM, however it comes at a cost of added complexity, and a strong need to understand how the signals do not violate the hold times.

*2.1.3 Full Array and Simulation*

After figuring out the proper design for the memory block and running the simulations necessary to convince the team that the design was robust and worked fairly well, they moved on to integrating everything together. The first thing they had to do was develop circuitry to generate all of the signals since they were still using the Analog Environment in cadence to develop them. The team decided that a clever way of implementing the generated signals would be to use a combination of different circuitry. Starting from the Read, Write, and Clock signals, two registers latch the values of Read and Write on the falling edge of the Clock. The purpose of this is so that these can then manipulate those signals needed throughout the rest of the circuitry. However, right after the registers for Read and Write, a multiplexer is needed for each signal; the Read and Write go into one input of their respective mux, while the other is connected to ground, and the value is selected by the Clock. This is done because the register latches the value at that node, and in order to have everything work properly for multiple cycles, the node after the register has to be reset to ground.

After these signals, there are several NANDs and NORs that alter the signals so they can be manipulated by the circuit. The signals then go through pulse generators to generate the precharge signals. The precharge generators take a signal that goes from one to zero and sends it to a NAND and through some inverters to generate a pulse. The pulse also goes onto some simple logic to develop the precharge signal, and to develop a intermediate word line signal. The signal is intermediate because it has to go to some simple logic with the 32 address bits to generate the 32 word line signals, where only one will be the word line signal, and the other 31 will be ground. Furthermore the sense amp enable signal also comes from the intermediate word line signal, only it goes into logic that makes it go low if write is on so that the sense amp only is on during a read. This is helpful because it gives the sense amp enable signal enough delay to look like a delayed word line, which is one common way of generating the sense amp enable signal. Finally, before being sent to the memory block, all the signals are sent through drivers so that they are strong enough to drive the logic.

Aside from generating all the necessary signals, there are several elements outside of the memory blocks that are necessary for the array to operate properly. There are two two-to-four decoders that take in the four high address bits, and these are used for choosing which of sixteen blocks to read from or write to. The way this is done is through sending them through simple logic with Read and Write signals before they are sent to the registers. The logic generates an enabled read and an enabled write bit which are the ones sent on to the blocks. This chooses the block because noop will be sent to all the blocks aside from the one that needs to be accessed. With the entire schematic complete, the team went on to run simulations to ensure that the cache ran properly.

* 1. **Testing**

In no way throughout the implementation of the design did simulations run smoothly; several steps throughout the way caused trouble. The first major instance of bad circuitry during simulation arose with the testing of the bit cell. Initially, it seemed as though the bit cell ran normally, and did not have any read upsets. However, after speaking with Calhoun the simulations had shown that given the right input vectors, the bit cell would not work properly. The team performed several tests at PVT corners to observe how to fix the bit cell. The knowledge gained from developing the bit cell was that extensive testing should be done with the worst case scenarios to show whether an entity works or not.

Other difficulties arose with verifying the efficiency of the sense amp. After testing several types of sense amplifiers, and finding that none of them were optimal for the high speed cache, a consultation with Calhoun presented a different sense amp that could work better. However, problems arose during simulations that made it appear as though the value read out was independent of the bit line values. The reason for this is that the timing of the signals was incorrect, and therefore threw off the readings in the sense amp. After research and testing, the signals were set to arrive in the way that upheld setup times and set up the sense amp for a successful read. The lessons earned from testing the sense amp were that setup and hold times must be upheld, and the easiest way of approximating them is through running simulations and observing where it breaks.

The other main issue was integrating all the circuit elements, because of several reasons. Simulations that seemed to hold true for simpler circuits fell apart as the circuits became more realistic and complete. The best example of this occurred while observing the change from a single memory block to the entire array. Despite that the simulation with a single block had approximations for the resistance and capacitance, they were not sufficient enough for the entire array, as that added a large amount of overhead. Then when it came to debugging the entire array, it became difficult to assess the errors in the schematic. Conclusions were that the only way to properly simulate when its on a smaller scheme is to not optimize at every level of integration, as that leads to an incorrect simulation at the next level of complexity.

Overall, when it comes to testing any schematic that has some valuable application, simply running basic simulations does not suffice. In order to have a good idea of robustness, one must run simulations where the input vectors could lead to a problem, simulate at the processing corners, different voltages or temperatures.

1. **RESULTS**

The high speed cache is a piece of technology that the group takes pride in. The group was unable to fully prove that the entire array worked, however made a lot of progress to show that they created a schematic that is close to operational. The group developed and tested two main schematics for the array. One of the schematics was based off the four corner model, and included approximate resistance and capacitance of one block. Attached to the block were all the components that would be used in order to access one block from outside in the array. This schematic was simulated and tested at all the process corners, at all temperatures, and at all voltages. The tests included reads followed by writes. All the tests in this schematic came back with positive results, and that the circuitry worked.

The team could not get the full 32 bit schematic to simulate a correct 32 bit read or write. For some reason, despite a large amount of work spent trying to fix it, the simulation refused to work. Intelligent attempts were made to get good signals to come out of the circuit, but nothing worked. Instead of outputting solid results, the team approximated values for the metrics based on educated guesses from previous simulations.

1. **DISCUSSION**Despite the fact that the current project is done, potential optimizations of this bit cell still exist. Under the presumption that we could continue forward with the project, we have a large amount of future plans. In the future, we plan to incorporate special features into the cache that will make the logic go faster. Special features such as current sense amplifiers and dynamic circuits could be implemented and possibly replace existing hardware.  
   Current sense amplifiers are special amplifiers that instead of directly measuring the voltage difference between two bit lines, they utilize current to decide what the output is. The other main difference from the differential voltage amplifier is that it has low impedance. Thus, the benefit of current sense amplifiers over differential voltage amplifiers is that it is faster and uses less power.   
   Furthermore, if the existing hardware would be replaced with more efficient hardware, or if some of the hardware gets replaced by dynamic logic, the delay of the cache could decrease. The benefit of including the dynamic logic is that most likely, the circuitry would consume less area, less power, and would be overall faster. This is because dynamic logic uses less transistors than the static circuits, and has a smaller pull up network capacitance to charge up. If given more time, it would seem like a feasible method to optimize the cache. Another objective that would be attempted would be connecting pads and designing the chip such that it could be manufactured.

http://www.cedcc.psu.edu/khanjan/sram.gif